Was bringt HTML5?

Dr. Thomas Meinike
Hochschule Merseburg (FH)

06.11.2009 – Wiesbaden

tekom Jahrestagung
mit tcworld conference
tcworld conference 2009

HoME
HOCHSCHULE
MERSEBURG
University of
Applied Sciences

HTML5?

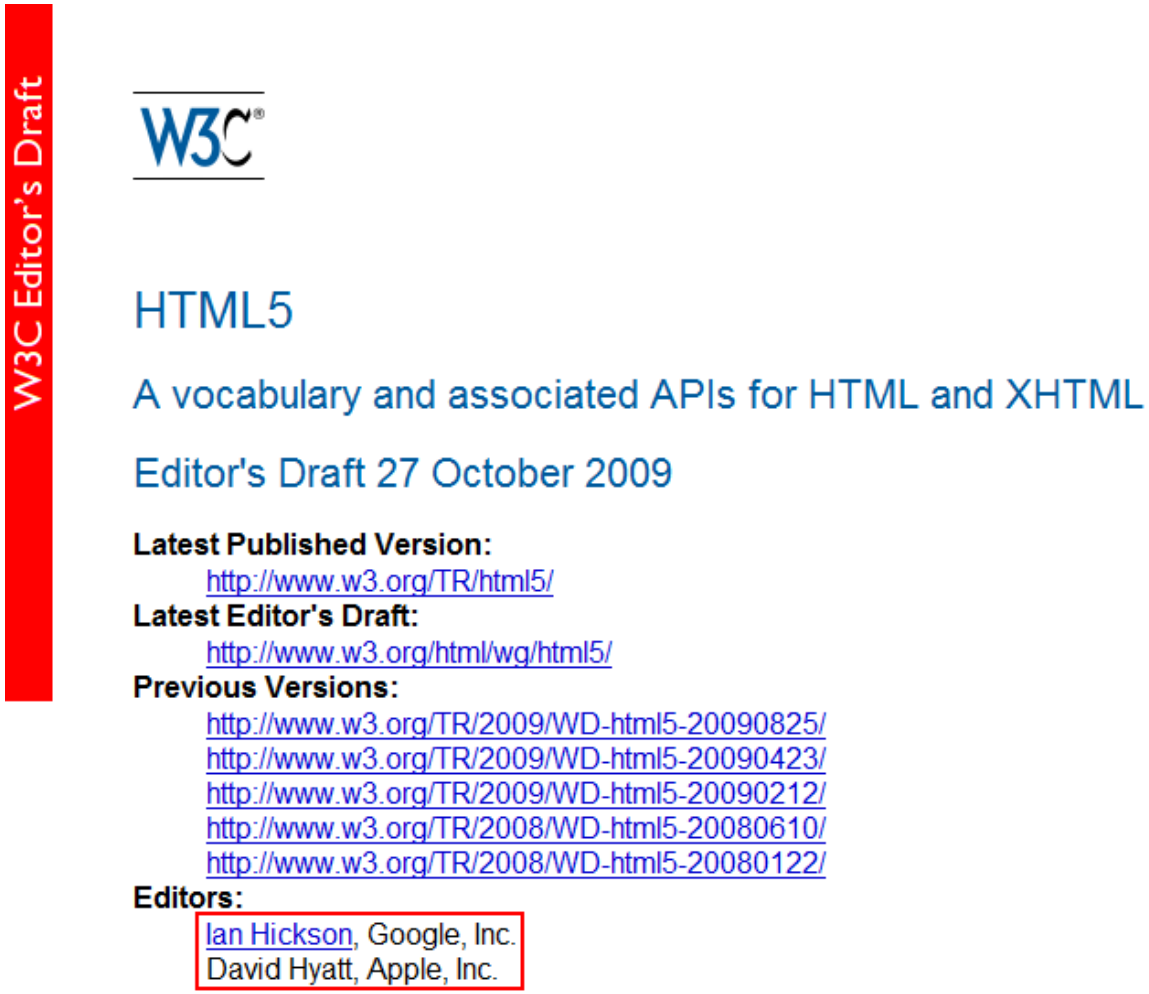
HTML5?

→ Einstieg (1)

- **HTML5** wurde ursprünglich seit 2004 von der außerhalb des W3C agierenden Arbeitsgruppe „WHATWG“ als Nachfolger von HTML 4.01 bzw. XHTML 1.x entwickelt
- Innerhalb dieser Gruppe versammelten sich Aktivisten von Apple, Google, Mozilla, Opera, …
- Als Arbeitstitel wurde „Web Applications 1.0“ gewählt, welcher bereits die (heutige) Zielrichtung angibt
- 2006/07 holt Tim Berners-Lee die „Abtrünnigen“ wieder unter das Dach des W3C, um eine HTML5-Arbeitsgruppe zu bilden
- Anfang 2008 lag der erste W3C-Entwurf vor
- 2009 wurden die „Web Forms 2.0“ Teil von HTML5
- Die parallel an XHTML 2 arbeitende Gruppe soll zum Ende 2009 aufgelöst werden
- 2010 soll bereits die finale HTML5-Empfehlung vorliegen

→ Einstieg (2)

- **HTML5 →** „A vocabulary and associated APIs for HTML and XHTML“



HTML5?

→ Einstieg (3)

- Schreibweise: **HTML5** (ohne Leerzeichen, seit 10/09)

- **Wesentliche Ziele:**
  - Kompatibilität zu existierenden Inhalten
  - Nutzung vorhandener Techniken
  - Evolution statt Revolution
  - Ansätze für Lösungen bisheriger Probleme (auf dem Weg vom Dokument zur Anwendung)
  - DOM-Konsistenz
  - Interoperabilität und Fehlertoleranz
  - Unterstützung möglichst offener Standards (Audio / Video)
  - Breite Unterstützung in Browsern und auf mobilen Geräten

HTML5

→ Allgemeines – Grundgerüst HTML5 vs. XHTML5

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="UTF-8">
    <title>...</title>
  </head>
  <body>
    <!-- weitere Inhalte -->
  </body>
</html>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
  <head>
    <title>...</title>
  </head>
  <body>
    <!-- weitere Inhalte -->
  </body>
</html>
```

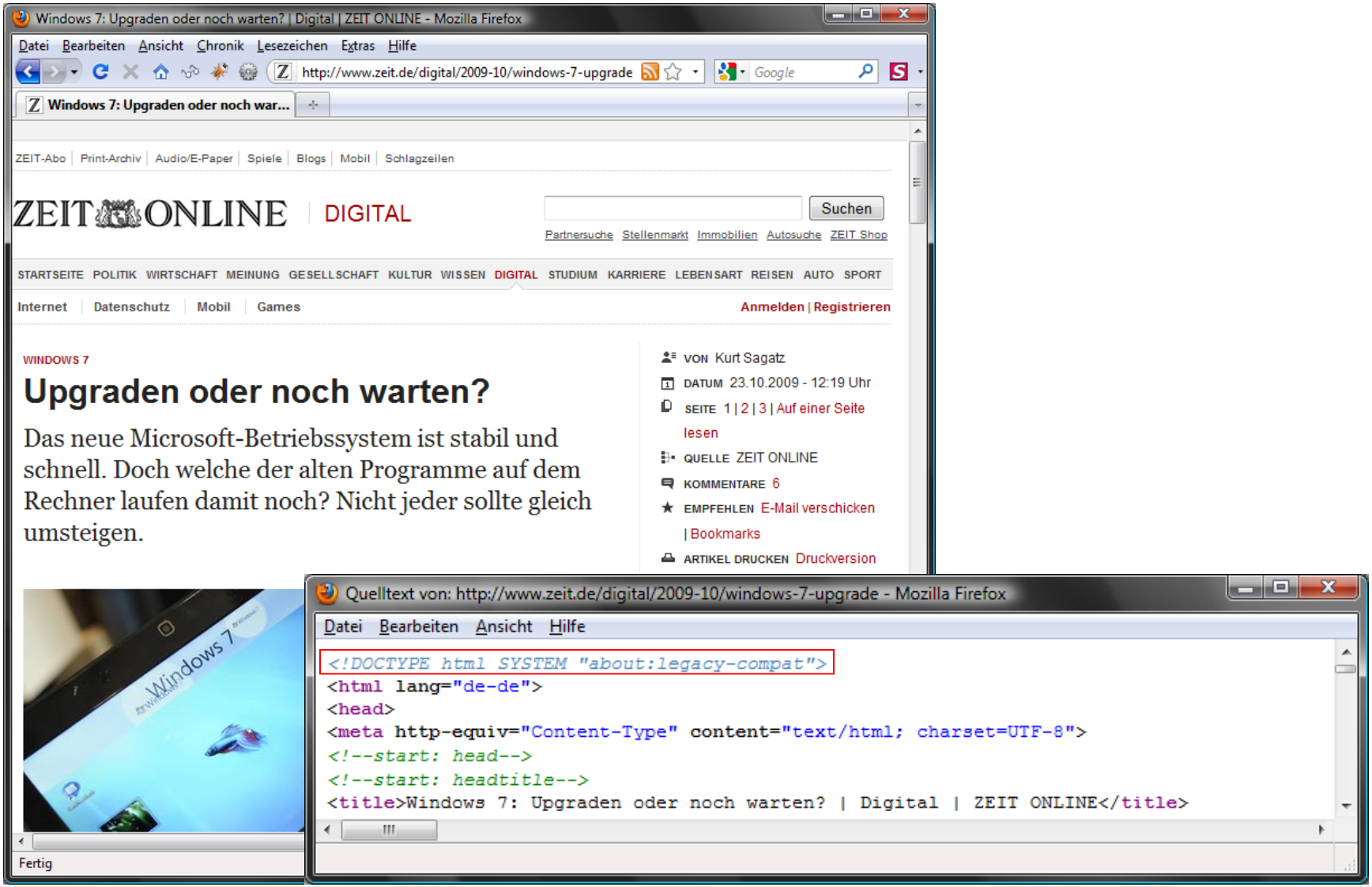
- **<!DOCTYPE html>** bedeutet (neben Einfachheit) vor allem: **keine DTD mehr**!
- DOCTYPE-Deklaration bestimmt lediglich den Browser-Modus (strict bzw. quirks)
- Für generierte Dokumente, etwa mittels XSLT, existiert eine alternative Schreibweise:

  <!DOCTYPE html SYSTEM “about:legacy-compat“>
- Vokabular, Grammatik und programmatischer Zugang wird über DOM-Spezifikation festgelegt
- (X)HTML-Code stellt die Serialisierung des DOMs dar
- Inhaltstypen / Parser: **text/html** für HTML5 bzw. **application/xhtml+xml** für XHTML5
- Validierung über **validator.w3.org** oder **html5.validator.nu** möglich

→ Allgemeines

- HTML5 erlaubt wieder mehr Freiheiten wie Weglassen optionaler schließender Tags (etwa **<p>Text** oder **<li>Text** sowie Attributwerte ohne Anführungszeichen und Standalone-Attribute (z. B. **<input type=radio checked>**)
- Erlaubt ist aber auch die bisherige XHTML-Notation, u. a. der abschließende Schrägstrich im Start-Tag bei leeren Elementen wie ****
- Bei der Verarbeitung von HTML auf der Basis von XML-Prozessen ist also keine Umstellung bzgl. der bisherigen Praxis nötig!
- HTML5 bietet neue Elemente für Struktur, Medieneinbindung und Spezialzwecke
- Gleichzeitig werden DOM-Schnittstellen für den Zugriff auf Struktur und Inhalte definiert und spezielle Anwendungsfelder erschlossen
- Bisherige Elemente aus den Strict-Versionen von HTML 4.01 / XHTML 1.0 bleiben bis auf **acronym** und **applet** erhalten, einige erhalten neue Bedeutung
- Neue Attribute werden eingeführt (u. a. für Formularelemente), frühere teilweise reaktiviert, z. B. <ol **start**="…">… (**ol** erhält neu **reversed**); das Attribut type wird für die Elemente style und script als optional eingestuft, **alt**-Attribut bei Bildern optional (aber sinnvoll)
- Somit kann ein (X)HTML-Dokument allein durch den neuen DOCTYPE bereits dem HTML5-Regelwerk genügen!

→ Allgemeines – Beispiel **zeit.de** (neuer DOCTYPE, noch keine neuen Elemente)

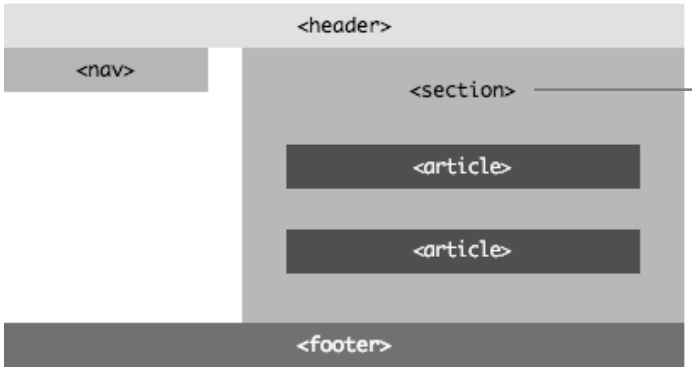


→ Allgemeines

- Die formale Bedeutung / Nutzung bisheriger Elemente wurde angepasst:
  - **a** als Platzhalter ohne href-Attribut zulässig
  - **a** darf nun auch Blockelemente enthalten: <**a** href="…"> Text </**a**>
  - **address** soll primär Kontaktangaben (im footer-Element) enthalten
  - **b** für stilistisch abweichende Texte (Schlüsselwörter)
  - **embed** wurde offiziell für Medieninhalte eingeführt, da bereits weit verbreitet
  - **hr** für Themenwechsel in einem Artikel oder Absatz
  - **i** für Texte mit geänderter Stimme / Stimmung (technische Begriffe, Idiome)
  - **label** legt Fokus nicht mehr auf das zugehörige Formularelement, sofern das nicht Systemstandard ist
  - **menu** soll für Toolbars und Kontextmenüs eingesetzt werden
  - **small** soll „Kleingedrucktes" auszeichnen (Kommentare, rechtliche Hinweise)
  - **strong** für wichtige Inhalte (nicht mehr für starke Betonung, betont wird mit **em**)

→ Neue Strukturelemente (1)

- **header** und **footer** für Kopf- und Fußbereiche eines Dokuments oder eines Artikels
- **article** für eigenständige Artikel (z. B. in Blogs), kann wiederum header, footer, … enthalten
- **section** fungiert als Anschnitt innerhalb eines Artikels, nicht als generischer Container (wurde 09/09 neu spezifiziert, insofern ist das Bild im Tagungsband bereits veraltet):



<section> soll hier durch ersetzt werden, d. h. es existiert kein generisches Element für den Hauptteil einer Struktur!

- **nav** als Containerelement für Navigation, typisch in Listenform (nav → ul → li)
- **aside** kann für Marginalien innerhalb eines Artikels oder als Sidebar („Blogroll“) eingesetzt werden
- **hgroup** zur Gruppierung mehrerer Überschriften (z. B. h1 + h2)
- **figure** wird u. a. als Block für Abbildungen mit Bildunterschrift eingeführt

→ Neue Strukturelemente (2)

```
...
<body>
  <header>
    <h1>Hauptüberschrift</h1><!-- sowie Logo, Suchfeld, ... -->
  </header>
  <!-- Hauptnavigation -->
  <nav>
    <ul><li>...</li><li>...</li><li>...</li></ul>
  </nav>
  <div id="content">
    <article>
      <header>
        <h1>Artikeltitel</h1>
        <p>Artikeleinführung ...</p>
      </header>
      <section>
        <h2>Abschnittstitel</h2>
        <p>Text ...</p>
        <p>Text ...</p>
      </section>
      <!-- ggf. weitere section-Elemente ... -->
      <!-- ggf. footer zum aktuellen Artikel -->
    </article>
    <!-- ggf. weitere article-Elemente ... -->
  </div>
  <aside><!-- optionale Sidebar --></aside>
  <footer>Kontakt, Impressum, weitere Links, ...</footer>
</body>
...
```

→ Neue Strukturelemente (3)

- **hgroup** → Überschriftengruppe, kann mit CSS gesondert formatiert werden:

```
<hgroup>
  <h1>Titel</h1>
  <h2>Untertitel</h2>
</hgroup>
```

- **figure** → Kindelemente **dt** für den Inhalt (hier Bild) und **dd** für Beschreibung (analog zur Definitionsliste **dl**):

```
<figure>
  <dt><img src="bildname.png" alt="Alternativtext" /></dt>
  <dd>Abb. 1: Bildinformation</dd>
</figure>
```

- **Blockbildung** für die Verwendung mit CSS in aktuellen Browsern (IE benötigt JS-Hilfe):

```
article, aside, figure, footer, header, hgroup, nav, section
{
  display: block;
}
```

```
<!--[if IE]><script type="text/javascript">
  document.createElement("article");
  // ...
  document.createElement("section");
</script><![endif]-->
```

→ Weitere neue Elemente (1)

- **details** → (ausklappbare) Zusatzinformationen oder Bedienelemente – ohne Angabe von **open** geschlossen, via **onclick**-Handler kann mit JavaScript interaktives Auf- / Zuklappen erreicht werden:

```
<details open="open"><!-- alternativ nur open -->
  <dt>Zusammenfassung (Legende)</dt>
  <dd>Weitere Inhalte ... </dd>
</details>
```

- **mark** → Hervorhebung von Inhalten:

```
<p><mark>HTML5</mark> stellt neue Elemente und Techniken zur Verfügung.</p>
```

- **meter** → Skalare Maße in bestimmten Bereichen oder anteilige / prozentuale Wertangaben (optionale Attribute: **high**, **low**, **max**, **min**, **optimum**, **value** – der Elementinhalt kann auch beschreibender Text sein):

```
<p>Der Durchmesser des Zylinders beträgt <meter min="5" max="20" value="10">10cm</meter>.</p>

<p>Die Wahlbeteiligung lag bei <meter>72.2%</meter>.</p>

<meter value="0.5">mittlere Intensität</meter>
```

→ Weitere neue Elemente (2)

- **menu** → Realisierung von Menüs / Toolbars für Webanwendungen (Kindelement **command** z. B. für Symbolmenüs bzw. verschachtelte **menu**-Elemente als Liste – siehe Attribute **type**, **label**, **icon** und JavaScript-Steuerung mit **onclick**-Handler):

```
<menu type="toolbar"><!-- contextmenu, list (Standardwert), toolbar -->
  <command label="Text_a" icon="a.png" onclick="Funktion()" />
  <command label="Text_b" icon="b.png" onclick="Funktion()" />
  <command label="Text_c" icon="c.png" onclick="Funktion()" />
</menu>
```

```
<menu type="list">
  <li>
    <menu label="Datei">
      <button onclick="Funktion()">Neu...</button>
      <button onclick="Funktion()">Laden...</button>
      <button onclick="Funktion()">Speichern...</button>
    </menu>
  </li>
  <li>
    <menu label="Hilfe">
      <button onclick="Funktion()">Themen...</button>
      <button onclick="Funktion()">Info...</button>
    </menu>
  </li>
</menu>
```

→ Weitere neue Elemente (3)

- **progress** → (dynamische) Fortschrittsanzeigen, JavaScript-Steuerung über den neuen Event-Handler **onprogress**:

```
<progress max="500" value="100">
  <span id="prog_out">20%</span>
</progress>
```

- **time** → Auszeichnung von Datum / Uhrzeit – Angaben über **datetime**-Attribut oder als Elementinhalt:

```
<p>Der Vortrag beginnt am <time datetime="2009-11-06T11:15+01:00">6. Nov. um 11:15</time>.</p>
<p>Öffnungszeit: <time>09:00</time> bis <time>18:00</time> Uhr</p>
```

time im Kontext von **article** → pubdate-Angabe bezieht die **datetime**-Angabe auf das übergeordnete article-Element (alternativ <article **pubdate**="…">…</article>):

```
<article>
  vom <time datetime="..." pubdate="pubdate">...</time>
  <!-- weitere Inhalte -->
</article>
```

→ Weitere neue Elemente (4)

- **audio** → Einbindung (direkt mit **src**-Attribut oder über Kindelement **source**) und Steuerung von Audio-Quellen (spezielle Attribute sowie JavaScript-Methoden):

```
<audio src="sound.mp3" type="audio/mpeg" autoplay controls loop>
  <p>Hier ggf. noch Download-Link oder Flash-Alternative als Fallback einbinden.</p>
</audio>
```

```
<audio controls>
  <source src="sound.ogg" type="audio/ogg" />
  <source src="sound.mp3" type="audio/mpeg" />
  <source src="sound.wav" type="audio/x-wav" />
  <!-- Download-Link / Fallback -->
</audio>
```

XHTML-konform:
`controls="controls"`
usw.

Browserunterstützung (10/09):

Browser	MP3	OGG	WAV
Chrome	x	x	–
Firefox	–	x	x (ab 3.1)
Opera	–	–	x (ab 10)
Safari	x	–	x (ab 3)

Optik in Safari (Windows):

JavaScript-Zugriff:

```
if(obj.paused)obj.play();
else obj.pause();

if(obj.ended)eigeneFunktion();
```

→ Weitere neue Elemente (5)

- **video** → Einbindung / Steuerung von Video-Material – weitere Attribute möglich: **autobuffer**, **autoplay**, **controls**, **loop** (boolean) und **poster=**“Vorschaugrafik-URL“):

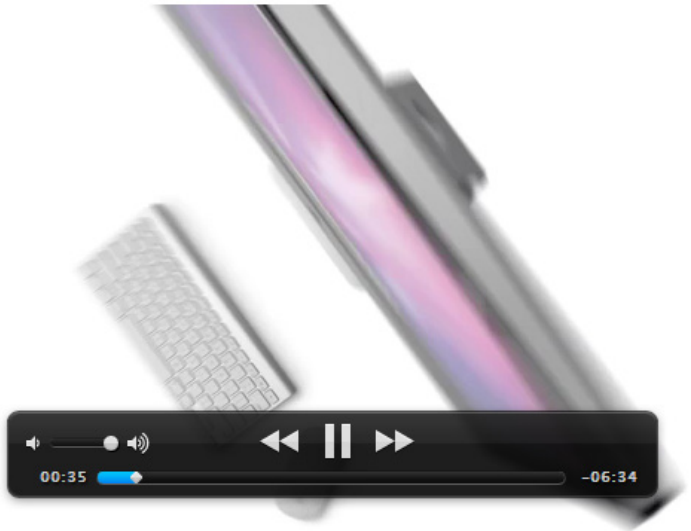
```
<video src="film.mp4" width="640" height="480" type="video/mpeg">
  <!--
    hier weitere Alternativen mit source-Element analog zu audio
    und Fallback-Angaben wie Download-Link
  -->
</video>
```

Browserunterstützung (10/09):

Browser	Formate
Chrome	H.264 (.mp4, .m4v) OGG (.ogg, .ogv)
Firefox und Opera	OGG
Safari	H.264

→ Bisher wurde keine Einigung zum Einsatz offener Audio-/Video-Codecs in den Browsern erreicht!

Apple-Werbevideo mit video-Element (dynamisch erzeugt + eigener Player): http://apple.com/imac/the-new-imac

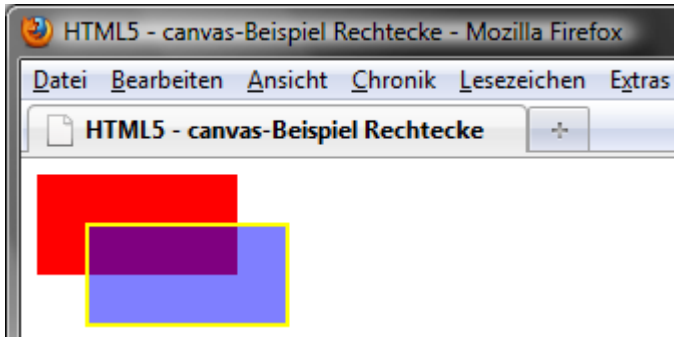


→ Weitere neue Elemente (6)

➢ **canvas** → **Bitmap**-Zeichenfläche mit adressierbaren Koordinaten für Rechtecke, Pfade und Rasterbilder mit Unterstützung von Text, Gradienten und Transformationen – Ansprache erfolgt vollständig mit JavaScript; im HTML-Code wird nur das **canvas**-Element mit ID sowie Breite und Höhe hinterlegt:

```
<canvas id="cv" width="640" height="480">
  <!-- Alternativinhalt, z. B. Rasterbild oder dargestellte Daten -->
</canvas>
```

```
window.onload=function()
{
  var canvas=document.getElementById("cv");

  if(canvas.getContext)
  {
    var context=canvas.getContext("2d");

    context.fillStyle="#F00"; // oder "rgb(255,0,0)"
    context.fillRect(0,0,100,50);

    context.fillStyle="rgba(0,0,255,0.5)"; // mit Alpha-Kanal 0...1
    context.fillRect(25,25,100,50);
    context.strokeStyle="yellow";
    context.lineWidth=2;
    context.strokeRect(25,25,100,50);
}
```

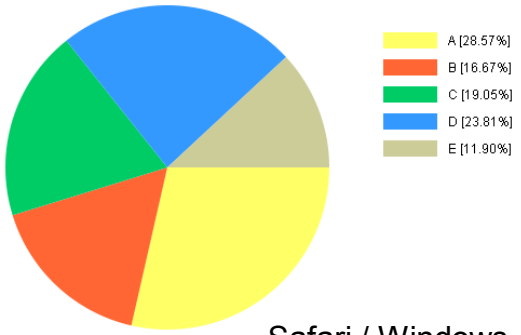


Hinweis: Die erzeugten Zeichenobjekte sind nicht im DOM verfügbar!

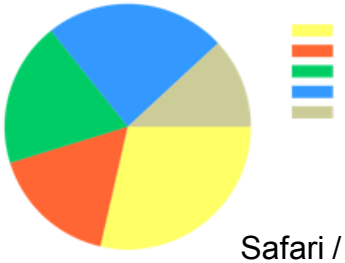
→ Weitere neue Elemente (7)

➢ **canvas** → Fortsetzung – Beispiel datenbasiertes Kreisdiagramm:

```
// Vorgaben der Daten, Farben (und Texte):
var daten=[120,70,80,100,50];
var farben=["#FF6","#F63","#0C6","#39F","#CC9"];

// Initialisierungen:
var anzahl=daten.length;
var summe=0;
var start_winkel=0,end_winkel=0;
for(var i=0;i<anzahl;i++)summe+=daten[i];

// canvas / context abfragen ...
// Daten aufbereiten
for(var i=0;i<anzahl;i++)
{
  // Kreissegment
  start_winkel=end_winkel;
  end_winkel+=daten[i]/summe*2*Math.PI;
  context.beginPath();
  context.moveTo(200,200);
  context.arc(200,200,150,start_winkel,end_winkel,false);
  context.closePath();
  context.fillStyle=farben[i];
  context.fill();
}
```



Safari / Windows



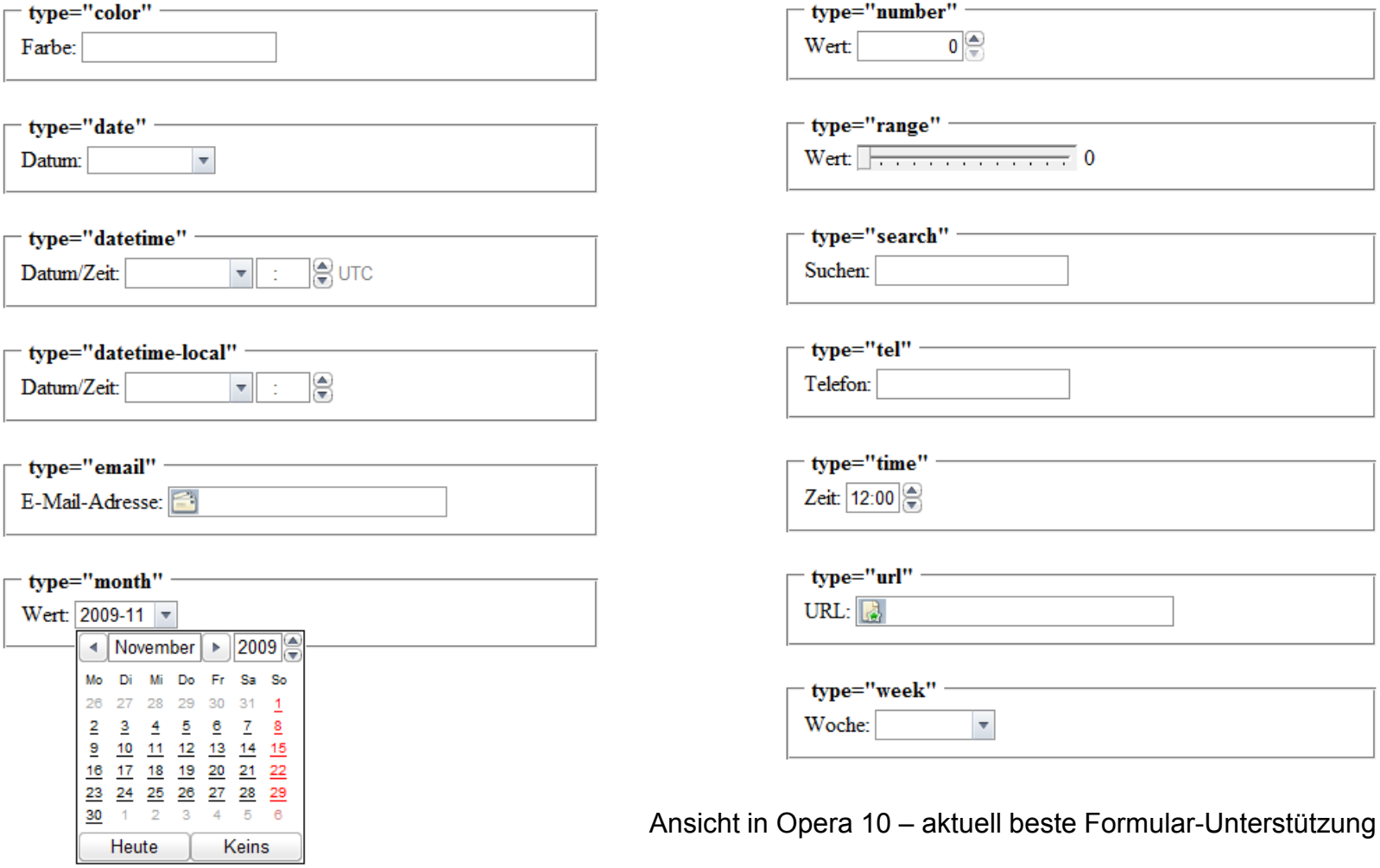
Safari / iPhone

```
<!--[if IE]>
  <script src="excanvas.js" type="text/javascript"></script>
<![endif]-->
```

IE-Unterstützung durch excanvas.js:
http://excanvas.sourceforge.net

→ Neue Formulartechniken (1)

➢ **Erweiterungen des type-Attributes** → mehr Auswahl- und Prüfmöglichkeiten:



Ansicht in Opera 10 – aktuell beste Formular-Unterstützung

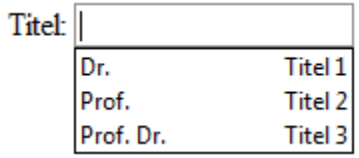
→ Neue Formulartechniken (2)

- Element **output** → Ausgabe von Ergebnissen, z. B. aktuelle Slider-Werte:

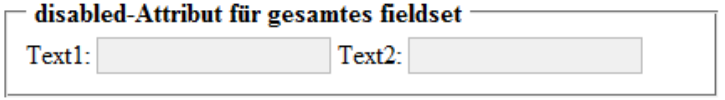
```
<input type="range" min="0" max="100" value="0" name="eingabe" />
<output name="ausgabe" onforminput="value=eingabe.value">0</output>
```

- Element **datalist** / input mit Attribut **list** → Eingabe bzw. Auswahl („Combo-Box“):

```
Titel: <input type="text" list="titelliste" />
<datalist id="titelliste">
  <option label="Titel 1" value="Dr." />
  <option label="Titel 2" value="Prof." />
  <option label="Titel 3" value="Prof. Dr." />
</datalist>
```



- **disabled**-Attribut für fieldset-Element:



- Weitere Attribute: **autofocus**, **multiple** (mehrere Dateien wählen mit type=“file“), **pattern** (reg. Ausdruck), **placeholder** (z. B. Suchbegriff …), **replace** (Text nach dem Absenden), **required** (Pflichtfeld)

- Formularelemente können außerhalb von <form>…</form> liegen und lassen sich mittels **form**-Attribut (sowie **formaction**, **formmethod**, …) benannten Formularen zuordnen

→ Microdata

- Festlegung von eigenen (sinnvollerweise vereinheitlichten) Key-/Value-Paaren für Mikroformatierungen – Attribut **itemscope** definiert eine Gruppierung von „items“, wobei jedes Paar eine Eigenschaft („item property“) bildet (Attribut **itemprop**):

```
<div itemscope>
  <p>Vortrag <span itemprop="thema"><em>Was bringt HTML5?</em></span>
    von <span itemprop="autor">Thomas Meinike</span></p>
</div>
```

- Mehrere Eigenschaften mit identischem Schlüsselwort, aber unterschiedlichen Werten, lassen sich ebenfalls zuweisen:

```
<div itemscope>
  <p>Hörenswerte Bands:</p>
  <ul>
    <li itemprop="band">Einstürzende Neubauten</li>
    <li itemprop="band">Nick Cave &amp;amp; The Bad Seeds</li>
    <li itemprop="band">Project Pitchfork</li>
  </ul>
</div>
```

- Programmatischer Zugriff soll via **Microdata DOM API** möglich sein: **document.getItems**(typeNames)

→ JavaScript-Erweiterungen zur Anwendungsentwicklung (1)

➢ **Geolocation-API** → Ermittlung des aktuellen Standpunktes:

```
window.onload=function()
{
  if(navigator.geolocation)
  {
    var geo=navigator.geolocation;
    geo.getCurrentPosition(getPosition,handleError)
  }
}

function getPosition(pos)
{
  var la=pos.coords.latitude;
  var lo=pos.coords.longitude;

  // Koordinaten verarbeiten, in Karte einbinden

  // außerdem zugänglich:
  // pos.coords.altitude und pos.coords.accuracy
  // pos.timestamp
}

function handleError()
{
  alert("Standort konnte nicht ermittelt werden!");
}
```

→ JavaScript-Erweiterungen zur Anwendungsentwicklung (2)

- **Web Storage API** → Speicherung von Daten (Key-/Value-Paare) alternativ zu Cookies; **localStorage** (bleibt erhalten) vs. **sessionStorage** (temporär für aktuelle Domain):

```
function setStorage()
{
  if(window.localStorage) // analog mit window.sessionStorage
  {
    window.localStorage.wert=document.inout.eingabe.value;
    // alternativer Zugriff: window.localStorage["wert"]
  }
}

function getStorage()
{
  if(window.localStorage) // analog mit window.sessionStorage
  {
    document.inout.ausgabe.value=window.localStorage.wert;
  }
}
```

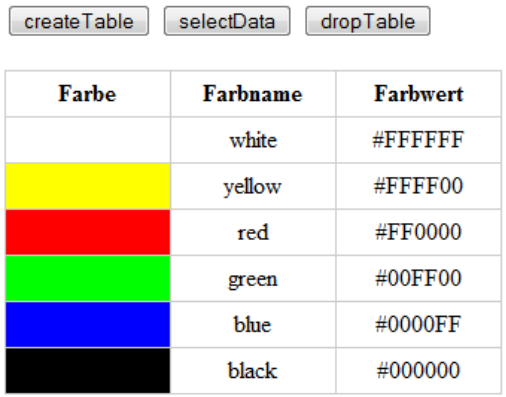
```
<form action="" name="inout">
  Set: <input type="text" name="eingabe" />
  Get: <input type="text" name="ausgabe" />
  <input type="button" value="setStorage" onclick="setStorage()" />
  <input type="button" value="getStorage" onclick="getStorage()" />
</form>
```

Safari / Windows

→ JavaScript-Erweiterungen zur Anwendungsentwicklung (3)

➢ **Web Database API** → Nutzung lokaler SQL-Datenbanken – **executeSql()**:

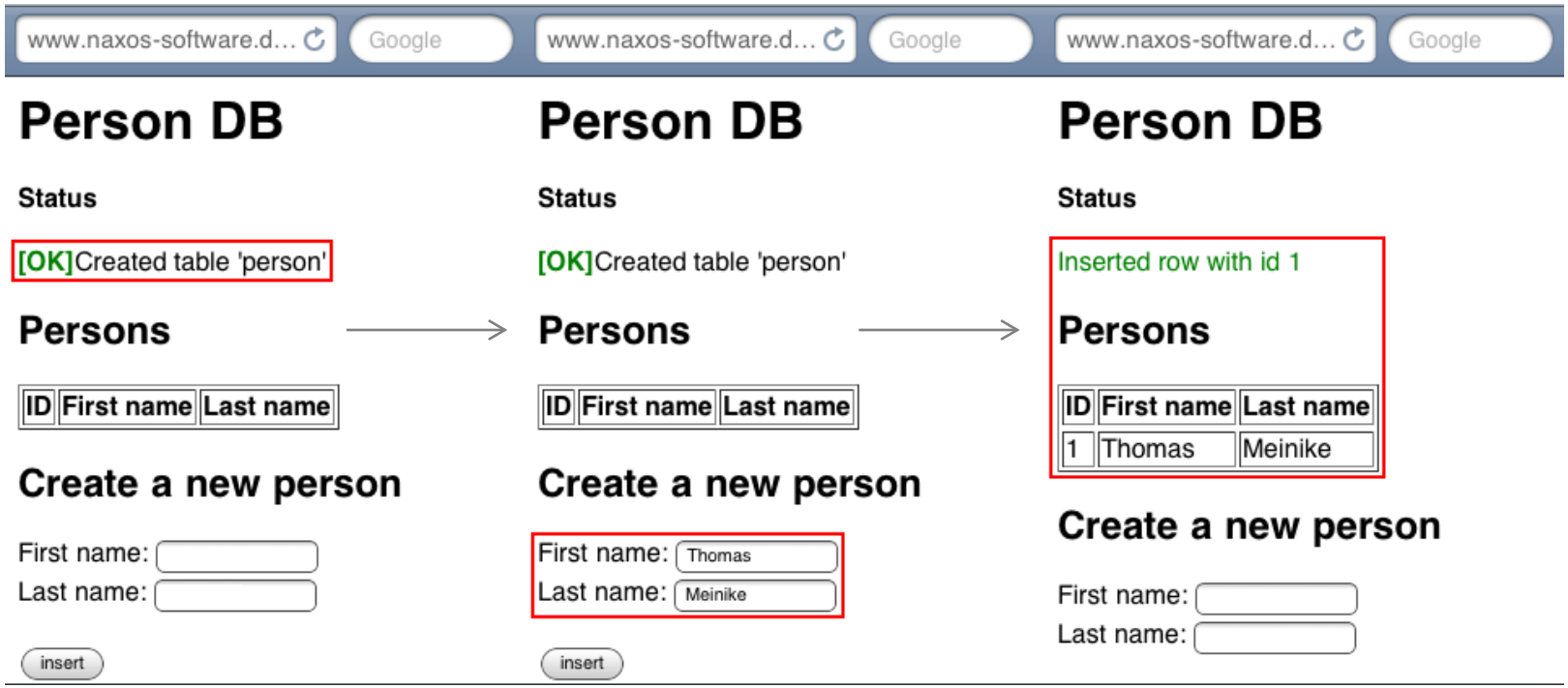
```
var dbref;

// Datenbank öffnen:
if(window.openDatabase) // aktuell nur Safari (WebKit)
{
  dbref=openDatabase("bspdb", "1.0", "Beispiel-DB", 100000);
}
else alert("DB-Funktionalität ist nicht verfügbar!");

// Tabelle in Datenbank erzeugen:
dbref.transaction(function(t)
{
  t.executeSql("CREATE TABLE IF NOT EXISTS farbtabelle
    (farbname TEXT,farbwert TEXT)", []);
});

// Daten in Tabelle einfügen:
dbref.transaction(function(t)
{
  for(var i=0;i<farbnamen.length;i++) // Daten aus zwei Arrays
  {
    t.executeSql("INSERT INTO farbtabelle (farbname,farbwert) VALUES (?,?)",
       [farbnamen[i],farbwerte[i]]);
  }
});

// analog Daten auslesen und verarbeiten: SELECT farbname,farbwert FROM farbtabelle ...
```



Safari / Windows

→ JavaScript-Erweiterungen zur Anwendungsentwicklung (4)

- **Offline Applications** → Webanwendungen u. a. für Mobiltelefone (Smartphones) – Einsatz der genannten Datenbank-Techniken mit **window.applicationCache** und neuen Events (**offline** / **online**) sowie CACHE MANIFEST für Offline-Dateien**;**
- Praktische Details enthält der Artikel zum online verfügbaren Naxos-Beispiel:



Naxos-Beispiel im Safari / iPhone

→ Sonstiges im Bereich JavaScript

- **Selectors API** → Zugriff auf Inhalte mit document.**querySelector**("...") bzw. document.**querySelectorAll**("...") unter Verwendung von CSS-Selektoren
- Neue Zugriffstechnik document.**getElementsByClassName**("...")
- document.**designMode** (on | off) und element.**contentEditable** (true | false) für editierbare Inhalte (etwa Rich-Text-Funktionalität)
- **Web Worker** für parallele Skriptprozesse, welche mit dem Hauptprozess interagieren
- **Web Sockets** für direkte Socket-Verbindungen mit Servern
- AJAX-Erweiterungen für **Cross-Domain-Kommunikation**
- **EventSource-Interface** für „Push Notifications" via HTTP
- … und weitere spannende Programmier- und Auszeichnungstechniken (etwa MathML und SVG direkt im HTML-Code einbinden) …

HTML5!

- Zusammenfassung / Ausblick
  - HTML5 wird mit besonderem Fokus auf Webanwendungen weiterentwickelt
  - Neue Elemente und Attribute bereichern vor allem die allgemeine Strukturauszeichnung, verbessern die Medieneinbindung und erweitern das Potenzial von Formularanwendungen
  - JavaScript wird zur clientseitigen Universalsprache – Entwicklern stehen neue leistungsfähige APIs zur Verfügung
  - HTML5 ist in aktuellen Browsern bereits ansatzweise einsetzbar (Chrome, Firefox, Opera und Safari) – alle haben unterschiedliche Stärken und Schwächen (Microsoft wartet auf noch eine finale Empfehlung)
  - Der Standardisierungsprozess liegt offenbar im Plan:
    „WHATWG sieht HTML5 auf der Zielgeraden“ (heise online, 28.10.2009)
  - Bei allem Fortschritt und Optimismus müssen auch die Aspekte der Zugänglichkeit stark JavaScript-lastiger Webanwendungen neu durchdacht werden
  - Schließlich kann man den aktuellen HTML5-Hype auch so sehen:
    „HTML5 ist die Fortsetzung des Browserkrieges mit anderen Mitteln.“
    (Mathias Schäfer alias molily im SELFHTML-Forum, 03.07.2009)

- Referenzen
  - Apple: “Safari Client-Side Storage and Offline Applications Programming Guide”; http://developer.apple.com/safari/library
  - Dive into HTML5: http://diveintohtml5.org
  - Diverse Artikel: http://alistapart.com | http://molily.de | http://webkrauts.de
  - HTML5 Doctor: http://html5doctor.com
  - Meinike, T.: http://datenverdrahten.de/html5
  - Naxos-Beispiele: http://www.naxos-software.de/blog
  - Stark, J.: “Building iPhone Apps with HTML, CSS, and JavaScript”; http://building-iphone-apps.labs.oreilly.com (Online-Text, O’REILLY-Buch 2010)
  - W3C: http://www.w3.org/TR/html5 und http://dev.w3.org/html5/spec
  - WHATWG: http://www.whatwg.org
- Kontakt
  - E-Mail: thomas.meinike@hs-merseburg.de
  - Twitter: http://twitter.com/XMLArbyter
  - WWW: http://www.iks.hs-merseburg.de/~meinike